
ELSEVIER

Data & Knowledge Engineering 39 (2001) 51–74

DATA & KNOWLEDGE ENGINEERING

www.elsevier.com/locate/datak

Supporting ontological analysis of taxonomic relationships

Christopher Welty [a,*], Nicola Guarino [b]

[a] *Computer Science Department, Vassar College, Poughkeepsie, NY 12604-0462, USA*
[b] *LADSEB-CNR, Padova, Italy*

Abstract

Taxonomies are an important part of conceptual modeling. They provide substantial structural information, and are typically the key elements in integration efforts, however there has been little guidance as to what makes a proper taxonomy. We have adopted several notions from the philosophical practice of formal ontology, and adapted them for use in information systems. These tools, *identity*, *essence*, *unity*, and *dependence*, provide a solid logical framework within which the properties that form a taxonomy can be analyzed. This analysis helps make intended meaning more explicit, improving human understanding and reducing the cost of integration. © 2001 Published by Elsevier Science B.V.

1. Introduction

Ontology is a discipline of Philosophy that deals with what is, with the kinds and structures of objects, properties, and other aspects of reality. While much of the philosophical practice of ontology dates back to Aristotle and what his students called ''metaphysics,'' the term *ontology* (ontologia) was coined in 1613 by Rudolf Gockel and apparently independently by Jacob Lorhard. According to the OED, the first recorded use in English was in 1721. Today's ontology includes questions such as, ''what is a castle?'' and ''what is a hole?'' The way we answer these questions reflects the way we perceive and interact with the world.

By the early 1980s, researchers in AI and especially in knowledge representation had realized that work in ontology was relevant to the necessary process of describing the world of intelligent systems to reason about and act in [25]. This awareness and integration grew and spread to other areas until, in the latter half of the final decade of the 20th century, the term ''ontology'' actually became a buzzword, as enterprise modeling, e-commerce, emerging XML meta-data standards, and knowledge management, among others, reached the top of many businesses strategic plans. In addition, an emphasis on ''knowledge sharing'' and interchange has made ontology an application area in its own right.

* Corresponding author. Tel.: +1-914-437-5992; fax: +1-914-437-7498.
E-mail address: weltyc@cs.vassar.edu (C. Welty).

0169-023X/01/$ - see front matter © 2001 Published by Elsevier Science B.V.
PII: S0169-023X(01)00030-1

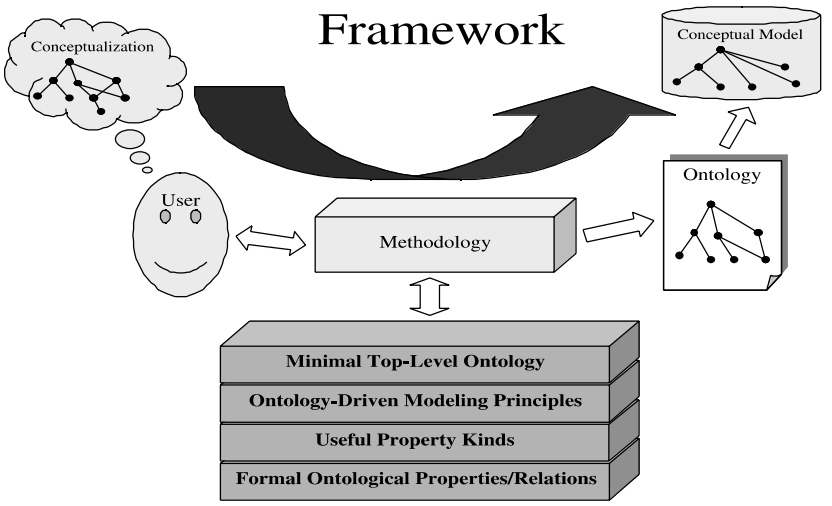

Fig. 1. Overview of the methodology.

In general, the accepted industrial meaning of ''ontology'' makes it synonymous with ''conceptual model'', and is nearly independent of its philosophical antecedents. We make a slight differentiation between these two terms, however (as shown later in Fig. 1): a conceptual model is an actual implementation of an ontology that has to satisfy the engineering trade-offs of a running application, while the design of an ontology is independent of run-time considerations, and its only goal is to specify the conceptualization of the world underlying such an application. In this paper we describe a well-founded methodology for ontological analysis (called OntoClean) that is strongly based on philosophical underpinnings, and a description-logic-based system that can be used to support this methodology. Although the methodology is not limited to analyzing taxonomies, we focus on that aspect of it here.

Most of the work described here have been published previously in a preliminary form, as will be noted in specific sections. This paper is an extended version of [13] that presents a broader view of the overall methodology and an extended discussion of a system to support it. We note that some valid criticisms of the formalizations we have presented previously have been raised in [19] and [6]. While the basic intuitions underlying these notions remain the same, some of the formalizations that follow have been updated in light of these criticisms. Further work and discussion are needed to fully address these changes and their implications, but it is out of the scope of this paper, whose main purpose is to provide an overview of the methodology we have developed for conceptual modeling.

2. Background

The notions upon which our methodology is based are subtle, so before describing them in more detail we discuss the basic intuitions behind them and how they are related to some existing notions in conceptual modeling.

2.1. Taxonomies

Taxonomies are a central part of most conceptual models. Properly structured taxonomies help bring substantial order to elements of a model, are particularly useful in presenting limited views of a model for human interpretation, and play a critical role in reuse and integration tasks. Improperly structured taxonomies have the opposite effect, making models confusing and difficult to reuse or integrate.

Clearly, insights into how to properly construct a taxonomy are useful. Many previous efforts at providing these insights have focused on the semantics of the taxonomic relationship (also called *is-a*, *class inclusion*, *subsumption*, etc.) [2], on different kinds of relations (*generalization*, *specialization*, *subset hierarchy*) according to the constraints involved in multiple taxonomic relationships (*covering*, *partition*, etc.) [28], on the taxonomic relationship in the more general framework of data abstractions [7], or on structural similarities between descriptions [1,4].

Our approach differs in that we focus on the arguments (i.e., the properties or concepts) involved in the subsumption relationship, rather than on the semantics of the relationship itself. The latter is taken for granted, as we take the statement "ψ subsumes ϕ" for arbitrary properties ψ and ϕ to mean that, necessarily: [1]

$$\forall x\ \Phi(x) \rightarrow \psi(x). \tag{1}$$

Our focus in this chapter will be on verifying the plausibility and the well-foundedness of single statements like (1) on the basis of the *ontological nature* of the two properties ϕ and ψ. Where for example description logics can determine whether one (complex) *description does* subsume another, this methodology can help determine whether or not a *primitive property can* subsume another.

2.2. Basic notions

We begin by introducing the most important philosophical notions: *identity*, *essence*, *unity*, and *dependence*. The notion of identity adopted here is based on intuitions about how we, as cognitive agents, in general interact with (and in particular recognize) individual entities in the world around us. Despite its fundamental importance in Philosophy, the notion of identity has been slow in making its way into the practice of conceptual modeling for information systems, where the goals of analyzing and describing the world are ostensibly the same.

The first step in understanding the intuitions behind identity requires considering the distinctions and similarities between *identity* and *unity*. These notions are different, albeit closely related and often confused under a generic notion of identity. Strictly speaking, identity is related to the problem of distinguishing a specific instance of a certain class from other instances of this class by means of a *characteristic property*, which is unique for *it* (that *whole* instance). Unity, on the other hand, is related to the problem of distinguishing the *parts* of an instance from the rest of the world by means of a *unifying relation* that binds the parts, and only the parts together. For example,

[1] All the ontological constraints we shall introduce here will be implicitly considered as necessary, i.e. true in every possible world. Indeed, we believe that modal necessity is what distinguishes – within a particular conceptualization of the world – an ontological truth from a contingent assertion. The specific modal logic adopted will be clarified later.

asking, "Is that my dog?" would be a problem of identity, whereas asking, "Is the collar part of my dog?" would be a problem of unity.

Both notions encounter problems when time is involved. The classical one is that of *identity through change*: in order to account for common sense, we need to admit that an individual may remain *the same* while exhibiting different properties at different times. But which properties can change, and which must not? And how can we reidentify an instance of a certain property after some time? The former issue leads to the notion of an *essential property*, on which we base the definition of *rigidity*, discussed below, while the latter is related to the distinction between *synchronic* and *diachronic* identity. An extensive analysis of these issues in the context of conceptual modeling has been made elsewhere [11].

The fourth notion, *ontological dependence*, may involve many different relations such as those existing between persons and their parents, holes in pieces of cheese and the cheese, and so on [27]. We focus here on a notion of dependence as applied to properties. We distinguish between *extrinsic* and *intrinsic* properties, according to whether they depend or not on other objects besides their own instances. An intrinsic property is typically something inherent in an individual, not dependent on other individuals, such as having a heart or having a fingerprint. Extrinsic properties are not inherent, and they have a relational nature, like "being a friend of John". Some extrinsic properties are assigned by external agents or agencies, such as having a specific social security number, having a specific customer ID, even having a specific name.

It is important to note that our ontological assumptions related to these notions ultimately depend on our *conceptualization* of the world [9]. This means that, while we shall use examples to clarify the notions central to our analysis, *the examples themselves will not be the point of this paper*. When we say, e.g., that "having the same fingerprint" may be considered an identity criterion for *PERSON*, we do *not* mean to claim this is the universal identity criterion for *PERSONs*, but that *if this were* to be taken as an identity criterion in some conceptualization, what would that mean for the property, for its instances, and its relationships to other properties?

To see how these intuitive notions can be used, consider for instance a bunch of bricks. The bricks are made and sit in a pile for a while, and are then used to build a castle. The castle, over time, crumbles back into a pile of bricks. How would we describe the lifetime of this bunch of bricks in terms of properties, relationships, and objects? Do we represent castles and bunches of bricks as two different properties? If so, do we have a single object which is always a "bunch of bricks", but which is only sometimes a "castle", or do we have two objects, one a castle and the other a bunch of bricks with a relationship between them? Our analysis helps with these choices by exposing certain assumptions underlying them. For example, many would agree that a bunch of bricks is identified by the bricks themselves – if we remove or replace a few bricks then we have a *different* bunch. A castle, on the other hand, does not seem to have this property – we can remove or add bricks to the castle and still consider that it is the *same* castle. Our framework is designed to clarify and exploit the logical consequences of decisions like this.

These decisions are ultimately the result of our sensory system, our culture, etc. and again the aim of this methodology is to clarify the formal tools that can both make such assumptions explicit, and reveal the logical consequences of them.

2.3. Related notions

Identity has many analogies in conceptual modeling for databases, knowledge bases, object-oriented, and classical information systems, however none of them completely captures the notion we present here. We discuss some of these cases below.

Membership conditions. In description logics, the conceptual models usually focus on the sufficient and necessary criteria for class *membership*, that is, recognizing instances of certain classes [3]. This is not identity, however, as it does not describe how instances of the same class are to be told apart. This is a common confusion that is important to keep clear: membership conditions determine when an entity is an instance of a class, i.e., they can be used to answer the question, "Is that *a* dog?" but not, "Is that *my* dog?"

Globally unique IDs. In object-oriented systems, uniquely identifying an object (as a collection of data) is critical, in particular when data are persistent or can be distributed [31]. In databases, *globally unique IDs* have been introduced into most commercial systems to address this issue. These solutions provide a notion of identity for the descriptions, for the units of data (individuals, objects or records), but not for the entities they describe. It still leaves open the possibility that two (or more) descriptions may refer to the same *entity*, and it is this entity that our notion of identity is concerned with. There is nothing, in other words, preventing two descriptions of the same dog from being created independently at different times/places, and thus having different IDs; The two records describe the same dog, but they are different pieces of data. Globally unique IDs provide identity criteria for database records, but not for the entities in the world the records describe.

Primary keys. Some object-oriented languages provide a facility for overloading or locally defining the equality predicate for a class. In standard database analysis, introducing new tables requires finding unique keys either as single fields or combinations of fields in a record. These two similar notions very closely approach our notion of identity as they do offer evidence towards determining when two descriptions refer to the same entity. There is a very subtle difference, however, which we will attempt to briefly describe here and which should become more clear with the examples at the end of the chapter.

Primary (and candidate) keys and overloaded equality operators are typically based on *extrinsic properties* that are required by a system to be unique. In many cases, information systems designers add these extrinsic properties simply as an escape from solving (often very difficult) identity problems. Our notion of identity is based mainly on *intrinsic properties*–we are interested in analyzing the inherent nature of entities and believe this is important for understanding a domain.

This is not to say that the former type of analysis never uses intrinsic properties nor that the latter never uses extrinsic ones – it is merely a question of emphasis. Furthermore, our analysis is often based on information which *may not be represented in the implemented system*, whereas the primary key notion can never use such information. For example, we may claim as part of our analysis that people are uniquely identified by their brain, but brains and their possession may not appear in the final system we are designing. Our notion of identity and the notion of primary keys are not incompatible, nor are they disjoint, and in practice conceptual modelers will need both.

3. The formal tools of ontological analysis

In this section we shall present a formal analysis of the basic notions discussed above, and we shall introduce a set of *meta-properties* that represent the behavior of a property with respect to these notions. Our goal is to show how these meta-properties impose some constraints on the way subsumption is used to model a domain, and to present a description logic system for checking these constraints.

3.1. Preliminaries

Let us assume that we have a first-order language L_0 (the modeling language) whose intended domain is the world to be modeled, and another first-order language L_1 (the meta-language) whose constant symbols are the predicates of L_0. Our meta-properties will be represented by predicate symbols of L_1. Primitive meta-properties will correspond to *axiom schemes* of L_0. When a certain axiom scheme holds in L_0 for a certain property, then the corresponding meta-property holds in L_1. This correspondence can be seen as a system of *reflection rules* between L_0 and L_1, which allow us to define a particular meta-property in our meta-language, avoiding a second-order logical definition. Meta-properties will be used as *analysis tools* to characterize the ontological nature of properties in L_0, and will always be defined with respect to a given conceptualization. In Section 5 we present an representation of L_0.

We denote primitive meta-properties by bold letters preceded by the sign "+", "−" or "∼", and the notation ϕ^M to indicate that the property ϕ has the meta-property M. The reading of each meta-property and its significance will be described in the subsequent sections.

In our analysis, we adopt a first-order logic with identity. This will be occasionally extended to a simple temporal logic, where all predicates are temporally indexed by means of an extra argument. If the time argument is omitted for a certain predicate ϕ, then the predicate is assumed to be time invariant, that is $\exists t\ \phi(x,t) \rightarrow \forall t\ \phi(x,t)$. Note that the identity relation will be assumed as time invariant: if two things are identical, they are identical forever. This means that Leibniz's rule holds with no exceptions.

We make some use in this paper of modal notions such as "necessary" and "possibly" operators which quantify over possible worlds; $\Box\phi$ means ϕ is *necessarily* true, i.e., true in all possible worlds, and $\Diamond\phi$ means ϕ is *possibly* true, i.e., true in at least one possible world. Our domain of quantification will be that of *possibilia*. That is, the extension of predicates will not be limited to what exists in the actual world, but to what exists in any possible world [22]. Therefore, we shall quantify over a constant domain in every possible world. Worlds will be considered "histories" rather than "snapshots", and we shall consider all of them as equally accessible. As a result, we shall adopt the simplest quantified modal logic, namely S5 plus the Barcan Formula [17].

For example, a predicate like "Unicorn" will not be empty in our world, although no instance has actual existence there. Actual existence is therefore different from existential quantification ("logical existence"), and will be represented by the temporally indexed predicate $E(x,t)$, meaning that x has actual existence at time t [16].

Finally, in order to avoid trivial cases in our meta-property definitions, we shall implicitly assume the property variables as restricted to *discriminating properties*, properties ϕ such that $\Diamond\exists x\ \phi(x) \wedge \Diamond\exists x\ \neg\phi(x)$. In other words, discriminating properties are properties for which there is

possibly something which has this property, and possibly something that does not have this property, they are neither tautological nor vacuous.

3.2. Rigidity

Lowe [23] defines an *essential* property of an object to be a property that is necessary for this object, i.e., the object has that property always and in every possible world. Essentiality is a relationship between an individual and a property. The notion of *rigidity* originally introduced in [8] is very much related to essentiality, and turns out to be very useful for conceptual modeling, This definition has evolved somewhat to reflect more accurately the way time and modality are related together:

Definition 1. A *rigid property* is a property that is essential to *all* its instances, i.e., a property ϕ such that: $\Box(\forall xt\ \phi(x,t) \rightarrow \Box\forall t'\ \phi(x,t'))$.

Definition 2. A *non-rigid property* is a property that is not essential to *some* of its instances, i.e., $\Diamond(\exists x,t\ \phi(x,t) \wedge \Diamond\exists t'\ \neg\phi(x,t'))$.

Definition 3. An *anti-rigid property* is a property that is not essential to *all* its instances, i.e., $\Box(\forall xt\ \phi(x,t) \rightarrow \Diamond\exists t'\ \neg\phi(x,t'))$.

For example, we normally think of *PERSON* as rigid; if x is an instance of *PERSON*, it must be an instance of *PERSON* in every possible world. The *STUDENT* property, on the other hand, is normally non-rigid; we can easily imagine an entity moving in and out of the *STUDENT* property while being the same individual.

Anti-rigidity was added as a further restriction to non-rigidity. The former constrains all instances of a property and the latter, as the simple negation of rigidity, is not very informative. Anti-rigidity attempts to capture the intuition that all instances of certain properties must possibly not be instances of this property. Consider the property *STUDENT*, for example: in its normal usage, every instance of *STUDENT* is not necessarily so.

Modal necessity is often confused with temporal permanence, but it is more general. For example, it is possible for a person to be a student their whole life, but this does not violate an anti-rigid assertion on the property *PERSON*. That the person was a student their whole life was accidental, i.e., not *necessary*: there are possible worlds in which this was not the case. The impact for information systems is simple: an anti-rigid property must *possibly* change, however there is no requirement that for each individual it does *actually* change. A rigid property, on the other hand, must not change. A non-rigid property has no restrictions, for some instances it may be essential (must not change), for some it may change. This weakness of non-rigidity led us to the stronger anti-rigid meta-property.

Rigid properties are marked with the meta-property $+\mathbf{R}$, non-rigid properties are marked with $-\mathbf{R}$, and anti-rigid properties with $\sim\mathbf{R}$. Note that rigidity as a meta-property is not "inherited" by sub-properties of properties that carry it, e.g., if we have $PERSON^{+\mathbf{R}}$ and $\forall x\ STUDENT(x) \rightarrow PERSON(x)$ then we know that all instances of *STUDENT* are necessarily instances of *PERSON*, but not necessarily (in the modal sense) instances of *STUDENT*. In this case, we would assert $STUDENT^{\sim\mathbf{R}}$ to indicate that every instance of *STUDENT* can cease to be a student, however no

instance of *STUDENT* can cease to be a person. On the other hand, we can show that anti-rigidity is inherited, e.g., any property subsumed by *STUDENT* must be anti-rigid.

Rigidity is also intuitively tied to existence, however we leave this philosophical discussion for future work.

3.3. Identity

In the philosophical literature, an *identity condition* (IC) for an arbitrary property ϕ is usually defined as a suitable relation ρ satisfying the following formula:

$$\phi(x) \wedge \phi(y) \rightarrow (\rho(x,y) \leftrightarrow x = y). \tag{2}$$

For example, an IC for the property *PERSON* may be *having-the-same-SSN or having-the-same-fingerprints*, if these relations are assumed to satisfy (2). When a relation ρ satisfying (2) exists for a property ϕ, we say the property *carries* an IC. The goal here is to find a relation that, for all instances of a property, is tantamount to identity. In information systems we typically provide artificial or *extrinsic* criteria for making this determination (q.v. Section 2.3), but in ontology this is the most basic question in explaining the existence of classes of entities: how do we know that an entity exists, how can it be identified?

As discussed in more detail elsewhere [11,15], the formulation in (2) has some problems, in our opinion. The first problem is related to the need for distinguishing between *supplying* an IC and simply *carrying* an IC: it seems that non-rigid properties like *STUDENT* can only carry their ICs, inheriting those supplied by their subsuming rigid properties like *PERSON*. The intuition behind this is that, since the same person can be a student at different times in different schools, an IC allegedly supplied by *STUDENT* (say, having the same registration number) may be only local, within a certain studenthood experience.

The second problem regards the nature of the ρ relation: what makes it an IC, and how can we index it with respect to time to account for the difference between *synchronic* and *diachronic* identity?

Finally, deciding whether a property carries an IC may be difficult, since finding a ρ that is both necessary *and* sufficient for identity is often hard, especially for natural kinds and artifacts. We believe therefore that a more practical approach is required for work in information systems.

For these reasons, we have defined (2) as follows:

Definition 4. An IC is a *sameness formula* Σ that satisfies either (3) or (4) below, excluding trivial cases [11] and assuming the predicate E for *actual existence* discussed in Section 3.1:

$$\Box(\mathrm{E}(x,t) \wedge \phi(x,t) \wedge \mathrm{E}(y,t') \wedge \phi(y,t') \wedge x = y \rightarrow \Sigma(x,y,t,t')), \tag{3}$$

$$\Box(\mathrm{E}(x,t) \wedge \phi(x,t) \wedge \mathrm{E}(y,t') \wedge \phi(y,t') \wedge \Sigma(x,y,t,t') \rightarrow x = y). \tag{4}$$

The *sameness formula* σ is a logical formula that expresses ICs for some property, usually in terms of the identity of related parts or properties. For example an IC for PERSON may be the formula $\mathit{fingerprint}(x,t) = \mathit{fingerprint}(y,t')$.

As correctly pointed out by Kaplan [19] and Carrara and Giaretta [5], there are many possible trivial formulae that satisfy the formal requirements for being an IC, but are not informative.

While we may begin to exclude logically some of these trivial cases, it is useful to point out that valid ICs must be informative.

An IC is necessary if it satisfies (3) and sufficient if it satisfies (4), and need not be both. This is, as mentioned above, weaker than (2), as is our reliance on actual existence, indicating that these formulae may only be *heuristics* for identity, however we believe they are more practical for information systems.

Based on this, we define two meta-properties:

Definition 5. Any property *carries* an IC iff it is subsumed by a property supplying this IC (including the case where it supplies the IC itself).

Definition 6. A property ϕ *supplies* an IC iff (i) it is rigid; (ii) there is an IC for it; (iii) the same IC is not carried by *all* the properties subsuming ϕ. This means that, if ϕ inherits different (but compatible) ICs from multiple properties, it still counts as supplying an IC.

Definition 7. Any property carrying an IC is called a *sortal* [30].

Any property carrying an IC is marked with the meta-property **+I** (**−I** otherwise). Any property supplying an IC is marked with the meta-property **+O** (**−O** otherwise). The letter "O" is a mnemonic for "own identity". From the above definitions, it is obvious that **+O** implies **+I** and **+R**. For example, both *PERSON* and *STUDENT* do carry identity (they are therefore **+I**), but only the former *supplies* it (**+O**). The property RED, e.g., normally does not carry identity – there is no necessary or sufficient formula that will identify or re-identify red things simply because they are red. Note that being red, the property itself, is trivially true of all instances of *RED* and is therefore not a legitimate IC.

3.4. Unity

In previous work we have extensively discussed and formalized the notion of unity, which is itself based upon the notion of part [11]. This formalization is based on the intuition that something is a whole if there exists a division such that all its members are connected to each other and to nothing else. We assume here that the axiomatization of the part relation is as shown in Table 1, where $P(x,y,t)$ means that x is a (proper or improper) part of y at time t.

Briefly, we define:

Definition 8. An object x *is a whole under* ω iff ω is a relation such that all the members of a certain division of x are linked by ω, and nothing else is linked by ω.

Table 1
Axiomatization of the part relation, adapted from Simons [27]

$\mathrm{PP}(x,y,t) =_{\mathrm{def}} P(x,y,t) \wedge \neg x = y$	Proper part
$\mathrm{O}(x,y,t) =_{\mathrm{def}} \exists z\ (P(z,x,t) \wedge P(z,y,t))$	Overlap
$P(x,y,t) \wedge P(y,x,t) \rightarrow x = y$	Anti-symmetry
$P(x,y,t) \wedge P(y,z,t) \rightarrow P(x,z,t)$	Transitivity
$\mathrm{PP}(x,y,t) \rightarrow \exists z\ (\mathrm{PP}(z,y,t) \wedge \neg \mathrm{O}(z,x,t))$	Weak supplementation

Definition 9. A property ϕ *carries a unity condition* iff there exists a single relation ω such that each instance of ϕ is *necessarily* a whole under ω.

Depending on the ontological nature of the ω relation, which can be understood as a "generalized connection", we may distinguish three main kinds of unity for concrete entities (i.e., those having a spatio-temporal location). Briefly, these are:

- *Topological unity*: based on some kind of topological or physical connection, such as the relationship between the parts of a piece of coal or an apple.
- *Morphological unity*: based on some combination of topological unity and shape, such as a ball, or a morphological relation *between wholes* such as for a constellation.
- *Functional unity*: based on a combination of other kinds of unity with some notion of purpose as with artifacts such as hammers, or a functional relation between wholes as with artifacts such as a bikini.

As the examples show, nothing prevents a whole from having parts that are themselves wholes (with a different UC). This can be the foundation of a theory of *pluralities*, which is however out of this paper's scope.

As with rigidity, in some situations it may be important to distinguish properties that do not carry a *common* UC for all their instances, from properties all of whose instances are not necessarily wholes. As we shall see, an example of the former kind may be *LEGAL AGENT*, all of whose instances are wholes, although with different UCs (some legal agents may be people, some companies). *AMOUNT OF MATTER* is usually an example of the latter kind, since none of its instances are wholes necessarily. Therefore we define:

Definition 10. A property has *anti-unity* if every instance of the property is not necessarily a whole.

Any property carrying a UC is marked with the meta-property **+U** (**−U** otherwise). Any property that has anti-unity is marked with the meta-property **∼U**, and of course **∼U** implies **−U**.

3.5. Dependence

The final meta-property we employ as a formal ontological tool is based on the notion of dependence. As mentioned in Section 2.2, we focus here on ontological dependence as applied to properties. The formalization below is based on Simons' definition of *notional dependence* [27]. We are aware that this is only an approximation of the more general notion of extrinsic (or relational) property, and that further work is needed (see for instance [18]).

Definition 11. A property ϕ is *externally dependent* on a property ψ if, for all its instances x, necessarily some instance of ψ must exist, which is neither a part nor a constituent of x:

$$\forall x \, \Box(\phi(x) \rightarrow \exists y \; \psi(y) \wedge \neg P(y,x) \wedge \neg C(y,x)). \tag{5}$$

The part relation P was discussed in Section 3.4. The relation $C(x,y)$ is used to denote *constitution*. Constitution differs subtly from part, in that it refers to the substance of which an entity is made. A castle is made of bricks, a statue from (perhaps) marble. Constitution usually relates concrete entities to mereologically essential wholes (i.e., collections or masses). Constitution is an

important notion to grasp, because it is commonly confused with subsumption. We discuss constitution with more rigor in [11], and give further examples of it in Section 6.

Clearly if we do not discount parts and constituents in (5), nearly all properties denoting classes of concrete entities would be dependent, since all non-atomic concrete entities have parts and are constituted of some material. In addition to excluding parts and constituents, a more rigorous definition must exclude qualities (such as colors), things which necessarily exist (such as the universe), and cases where ψ is subsumed by ϕ (since this would make ϕ dependent on itself). Intuitively, we say that, for example, *PARENT* is externally dependent on *CHILD* (one cannot be a parent without having a child), but *PERSON* is neither externally dependent on heart nor on body (because any person has a heart as a part and is constituted of a body).

An externally dependent property is marked with the meta-property **+D** (**−D** otherwise).

3.6. Constraints and assumptions

The first observation descending immediately from our definitions regards some *subsumption constraints*. If ϕ and ψ are two properties then the following constraints hold:

$$\phi^{\sim\mathbf{R}} \text{ must subsume } \psi^{\sim\mathbf{R}}, \tag{6}$$

$$\phi^{+\mathbf{I}} \text{ must subsume } \psi^{+\mathbf{I}}, \tag{7}$$

$$\phi^{+\mathbf{U}} \text{ must subsume } \psi^{+\mathbf{U}}, \tag{8}$$

$$\phi^{\sim\mathbf{U}} \text{ must subsume } \psi^{\sim\mathbf{U}}, \tag{9}$$

$$\phi^{+\mathbf{D}} \text{ must subsume } \psi^{+\mathbf{D}}, \tag{10}$$

Properties with incompatible ICs/UCs are disjoint. (11)

Constraints (6)–(10) follow directly from our meta-property definitions (see [12] for more discussion and examples), and (11) should be obvious from the above discussion of identity and unity, but it is largely overlooked in many practical cases [12,15]. See Section 6 for an example that shows the practical use of these constraints.

Finally, we make the following assumptions regarding identity (adapted from Lowe [23]):

- *Sortal individuation*. Every domain element must instantiate some property carrying an IC **(+I)**. In this way we satisfy Quine's dicto "No entity without identity" [26].
- *Sortal expandability*. If two entities (instances of different properties) are the same, they must be instances of a property carrying a condition for their identity.

4. Methodology

The specific goal of this methodology is to *make modeling assumptions clear*. One of the most important ways the methodology is used is in analyzing taxonomies to form *well-founded taxonomies*, which are discussed further in Section 6.

The methodology is made up of a number of formal analysis tools that can be grouped into four distinct layers, such that the notions and techniques within each layer are based on the

notions and techniques in the layers below. In Fig. 1, the methodology is depicted as four layers that support a process in which a person's or group's conceptualization evolves into a concrete conceptual model. In this section, we very briefly outline the purpose of each layer and the tools in it, followed by a brief discussion of a system to support the methodology.

4.1. First layer: Foundations

In the lowest, foundational, layer of the methodology are the meta-properties described in Section 3. As discussed there, the meta-properties correspond to axiom schemes in the modeling language and properties in the meta-language. Properties in the modeling language correspond to constant symbols in the meta-language. This proves important for our support system, which implements only the meta-language.

4.2. Second layer: Useful property kinds

The second layer in the methodology contains an ontology of useful property kinds, originally presented in [12].

The formal ontology of properties discussed in [12] distinguishes eight different kinds of properties based on the valid and most useful combinations of the meta-properties discussed in Section 3. These are shown in Table 2 and in Fig. 2. These property kinds enrich a modeler's ability to specify the meaning of properties in an ontology, since the definition of each property kind includes an intuitive and domain-independent description of how this kind of property should be used in an ontology.

Table 2
All possible combinations of the meta-properties

<table>
<tr><td>+O</td><td>+I</td><td>+R</td><td>+D</td><td>Type</td><td rowspan="5">Sortal</td></tr>
<tr><td></td><td></td><td></td><td>-D</td><td></td></tr>
<tr><td>-O</td><td>+I</td><td>+R</td><td>+D</td><td>Quasi-type</td></tr>
<tr><td></td><td></td><td></td><td>-D</td><td></td></tr>
<tr><td>-O</td><td>+I</td><td>~R</td><td>+D</td><td>Material role</td></tr>
<tr><td>-O</td><td>+I</td><td>~R</td><td>-D</td><td>Phased sortal</td><td></td></tr>
<tr><td>-O</td><td>+I</td><td>¬R</td><td>+D</td><td>Mixin</td><td></td></tr>
<tr><td></td><td></td><td></td><td>-D</td><td></td><td></td></tr>
<tr><td>-O</td><td>-I</td><td>+R</td><td>+D</td><td>Category</td><td rowspan="6">Non-sortal</td></tr>
<tr><td></td><td></td><td></td><td>-D</td><td></td></tr>
<tr><td>-O</td><td>-I</td><td>~R</td><td>+D</td><td>Formal Role</td></tr>
<tr><td></td><td></td><td>~R</td><td>-D</td><td>Attribution</td></tr>
<tr><td></td><td></td><td>¬R</td><td>+D</td><td></td></tr>
<tr><td></td><td></td><td></td><td>-D</td><td></td></tr>
<tr><td rowspan="3">+O</td><td>-I</td><td></td><td></td><td rowspan="3">incoherent</td><td rowspan="3"></td></tr>
<tr><td rowspan="2">+I</td><td>~R</td><td rowspan="2"></td></tr>
<tr><td>-R</td></tr>
</table>

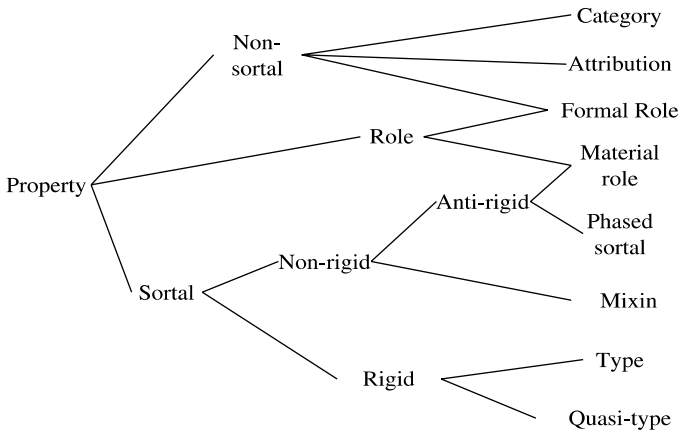

Fig. 2. Taxonomy of properties.

As discussed later, deciding what the identity criteria are for all instances of a property is a subtle task that requires training and experience, although engaging in discussion with team members on these questions is often, in practice, very illuminating. Given that our methodology only requires knowing whether a property has an IC or not, our system contains examples of common kinds of identity criteria as ad hoc meta-properties, to assist less experienced modelers with these decisions. For example, the meta-property **+CO**, designates a *countable* property. This is an important specialization of sortals. In many cases, besides carrying identity (**+I**), countable properties also carry unity (**+U**). A countable property can only subsume other countable properties, and indicates that its instances can be counted. For example, the property *WATER* is ordinarily not countable (one does not count the water filling a cup), however the property *CUP* is countable. Of course, this does not imply that the property *WATER* does not have identity; a property may be non-countable and still have an IC.

4.3. Third layer: Ontology-based modeling principles

The third layer in the methodology contains the notions of *backbone property* and *stratification*.

The backbone taxonomy. One of the principal roles of taxonomies is to impart structure on an ontology, to facilitate human understanding, and to enable integration. We have found that a natural result of our analysis is the identification of special properties in a taxonomy that best fill this role. We call these properties *backbone properties*, which constitute the backbone *taxonomy* [12].

The backbone taxonomy consists only of rigid properties, which are divided into three kinds (as discussed above): *categories*, *types*, and *quasi-types*. Categories cannot be subsumed by any other kinds of properties, and therefore represent the highest level (most general) properties in a taxonomy. They are usually taken as primitive properties.

Types are critical in our analysis because according to the assumptions presented in Section 3.6, *every instance instantiates at least one of these properties*. Therefore the backbone provides the basic structure of the entire domain.

These notions give rise to an idealized view of how ontologies should be structured taxonomically, as shown in Fig. 3. While strict adherence to this idealized structure may not always be possible, we believe that following it to the degree possible will grow to be an important design principle for conceptual modeling, with pay-offs in understandability and ease of integration.

Stratification. A very important result of our analysis is the recognition of multiple entities, based on incompatible identity or unity criteria, where usually only one entity is conceived. The classical example is the statue and the clay it is made of, which count as different objects in our analysis. This view results in a *stratified ontology* [10], where entities belong to different levels, depending on their unity and identity assumptions:

- **+ME.** Properties carrying a *mereologically extensional* IC. Certain properties, as discussed in Section 3.5, concerning masses or plural entities have as a necessary identity condition that the parts or members of their instances must be the same (instances cannot change their parts). For example *GROUP-OF-PEOPLE*, if the members change, it is a different group. These properties cannot subsume properties with **−ME**.
- **+UT.** Properties carrying *topological* unity. See Section 3.4. Properties with **+UT** have unity (**+U**), and cannot subsume properties with **−UT**. In addition, these properties are normally dependent on properties with **+ME**, due to constitution. A brick, for example, is constituted of clay.
- **+UM.** Properties carrying *morphological* unity. See Section 3.4. Properties with **+UM** have unity (**+U**), and cannot subsume properties with **−UM**. In addition, these properties are normally dependent on properties with **+UT**, due to constitution. A garden path, for example, may be constituted of non-touching bricks scattered in a particular pattern.
- **+UF.** Properties carrying *functional* unity. See Section 3.4. Properties with **+UF** have unity (**+U**), and cannot subsume properties with **−UF**. In addition, these properties are normally dependent on properties with **+UT** or **+UM**, due to constitution. A castle, for example, may be constituted of bricks, and is only a whole while it can still function as a castle.

We distinguish, based on these meta-properties, levels of ontological stratification: the physical level, the functional level, etc. In addition, other types of identity and unity conditions can be conceived that would define e.g., the biological level (living things), the social level (organiza-

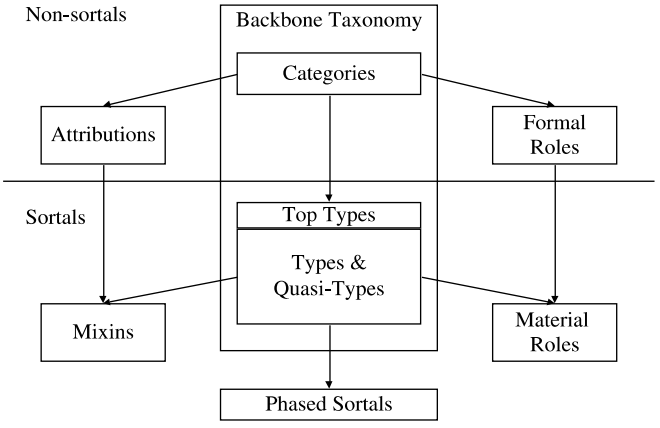

Fig. 3. Ideal taxonomy structure. In general, a property in one group must be subsumed by a property in the group(s) above it.

tions), etc. Entities at the higher levels are *constituted* (and co-located with) entities at the lower levels. Properties at each level are disjoint from properties at other levels. The advantage of this view is a better semantic account of the taxonomic relation, a better account of the hidden ontological assumptions, and in general better ontologies. The costs are a moderate proliferation (by a constant factor corresponding to the number of levels) of the number of entities in the domain, and the necessity to take into account different relations besides *is-a*, such as *dependence*, *spatio-temporal co-localization*, and *constitution*.

4.4. Fourth layer: Top level ontology

The highest layer of our methodology is a top-level ontology designed using the notions and techniques of the layers below. First steps towards this layer of the methodology have been discussed in [10].

4.5. Question/answer system

Finally, we have captured the notions and techniques from these four layers in a knowledge-based system that guides conceptual modelers through the analysis process. This approach is similar to that of [29], and is described more fully in Section 5. The system implements only the meta-language, providing some consistency checking of the constraints outlined in Section 3.6.

5. Knowledge-based support

The methodology based on these techniques requires that the assignment of meta-properties to properties in an ontology be performed by hand. This analysis in all cases requires that the modeler be very clear about what each property *means*. We have developed a support system that can help modelers with this analysis, as it can verify the consistency of a taxonomy based on the constraints described in Section 3.6. A modeler enters information about properties to be used in a conceptual model, and the proposed taxonomic structure. Meta-properties are assigned, and the consistency of the taxonomy is then checked automatically.

In this section we describe aspects of the system and in the next section walk through a simple example using the system.

5.1. Overview

The system implements all the constraints and all the inferences described in previous sections for the meta-language. It has two basic modes of operation: a question/answer (Q/A) mode and a batch mode. It is our intention to make both modes of the system available online.

In Q/A mode, the system is designed to assist a modeler in choosing the appropriate meta-properties for their properties by asking a series of questions. More general questions are asked first, such as "Does the property carry identity?" and the modeler may respond yes, no, or unsure. If unsure about a meta-property, more specific questions can be asked such as, "Are instances of the property countable?" It is always possible to leave answers unknown, of course in these cases the system cannot verify the correctness of those properties.

As the properties are being entered, the information presented so far is checked for consistency, and any inferences are made. For example, if a modeler answers "yes" to the question that assigns the $\sim$**R** meta-property, the system will infer that the property is also $-$**R**. If a modeler assigns **+R**

to a property and also asserts that it is subsumed by a property with $\sim$**R**, the system will immediately raise an appropriate error.

The system is implemented in CLASSIC, a description logic system developed at AT&T Bell Labs in 1990 [3]. CLASSIC was chosen mainly because of its familiarity to the implementors, however there are some good rationale for this choice. All the reasoning required of the meta-language is provided, almost no code other than I/O was needed. CLASSIC is the only description logic with a full explanation system implemented and included, making it possible to generate explanations for constraint violations (as opposed to simply saying there was a violation).

The total system without the meta-property definitions is under 14K of LISP code. The system works by first loading in the definitions, allowing us to add and modify them. Then the modeler may invoke the Q/A system or simply load a batch file consisting of properties and their meta-property tags (**+R**, **+I**, etc.). An example of how the meta-properties are defined is given in Fig. 4, as well as an example property specification for batch mode.

5.2. Reasoning

The reasoning required of the system is fairly rudimentary for a description logic. In addition to the obvious implications of the property kind taxonomy shown in Fig. 2 (i.e., a type is a sortal and a rigid sortal), the system is designed so that no redundant information need be specified, making the job of the modeler a bit easier.

The system uses the open world assumption regarding the meta-properties of the properties the modeler enters. A property is not assumed to have any meta-properties until they are asserted by the modeler. To accomplish this within the description logic framework, as well as provide for the possibility of "unknown" answers in a binary truth valued logic, *each possible* meta-property assignment is represented as a concept. For example, there is a concept corresponding to the rigid meta-property, as well as concepts for non-rigid and anti-rigid, i.e., three concepts as shown in Fig. 4. Opposite meta-properties are handled by making their corresponding concepts disjoint from each other, thus when you assert a property is e.g., rigid, the system knows that it cannot be non-rigid. Anti-rigid is asserted to be subsumed by non-rigid, thus when a property is assigned $\sim$**R** it is known to be $-$**R** as well. To leave a meta-property as unknown, the modeler must basically answer "no" to two questions, e.g., "Is the property rigid?" and "Is the property non-rigid?"

It is important to keep in mind that since the properties of the modeler's ontology are actually constant symbols in the meta-language, the modeler's properties are represented as individuals in the system, and are instantiated when the modeler enters their names. They are allowed to have two relations: the generalization/specialization-of relation and the name of a characteristic relation. For each relation, the modeler is asked if there are values for it.

```
(define-meta-prop rigid-property nil
  :disjoint-partition rigid
  :tag "+R"
  :classify-message "The property ~a is rigid"
  :question "Is this property rigid?")

(define-meta-prop non-rigid-property nil
  :tag "-R"
  :classify-message "The property ~a is non-rigid"
  :question "Is this property non-rigid?"
  :disjoint-partition rigid)
```

```
(define-meta-prop anti-rigid-property
  non-rigid-property
  :tag "~R"
  :disjoint-partition anti-rigid
  :classify-message "The property ~a is anti-rigid"
  :question "Is this property anti-rigid?")

(define-property 'red-apple :ask? nil
  :tags "+I-O+U-D~R"
  :rvs '((subsumed-by apple red)))
```

Fig. 4. Example meta-property definitions.

Since the modeler's properties are individuals, the description logic does not provide any special reasoning services for subsumption between them. Classic provides automatic relation inverses, so asserting that one property is a specialization of another causes the generalization inverse to be asserted as well. In addition, the generalization/specialization relations between the individuals are defined to be transitive, as expected. This requires some special machinery since Classic does not support transitivity, but can easily be accomplished in the standard way with a primitive (non-transitive) version of the relation and the transitive version, as with the canonical parent/ancestor relations in Prolog.

All the constraints (other than the obvious ones provided by disjointness) are expressed as necessary conditions on the concepts representing the meta-properties. For example, constraint (6) in Section 3.6 is represented as a necessary condition on anti-rigid properties: $\forall$ *generalization-of* $\cdot$ *ANTI-RIGID-PROPERTY* (i.e., an anti-rigid property can only be the generalization of anti-rigid properties).

All inference and constraint checking is done as soon as the information is made available by the modeler. The explanation system provided by Classic is therefore particularly important in batch mode, as the modeler cannot benefit from the context of having just answered a question to know why an inconsistency was generated.

5.3. Evaluation

While the system was not designed to be particularly usable, we have evaluated it along two dimensions: effectiveness of the questions and scalability.

5.3.1. Scalability

We have tested the system on randomly generated hierarchies up to 20,000 nodes in batch mode. The reasoning the system performs is trivial and experimental results indicate two dimensions of complexity: number of properties and number of parents. In each dimension, complexity was observed to be linear, with the system performing all reasoning as fast as the batch files would load. Combining the two dimensions (i.e., large datasets with much multiple inheritance) resulted in polynomial increases.

We are not concerned with the latter result as the artificial data were difficult to generate, and based on our experiences do not seem to correspond to real systems. In fact, we believe an important result of our methodology is a drastic reduction in multiple inheritance links between properties [12].

5.3.2. Effectiveness of questions

We have found that among people who have done a lot of conceptual modeling, many aspects of our methodology make sense. In fact the methodology is in use by several companies for conceptual modeling and integration projects such as OntoWorks and Document Development Corporation, among others. The main difficulty in applying it, however, is understanding when and what identity and unity conditions apply to properties in a domain.

We have attempted to gather together a few examples of common identity and unity criteria, and the system is designed to incorporate this additional information as further sub-concepts of the existing meta-property definitions.

When a modeler is unsure about a particular meta-property, and therefore answers "no" to the questions for the two disjoint concepts that represent it, the system moves further down the hierarchy of concepts. If a modeler is sure about a meta-property and therefore answers "yes" to a question, the Q/A system does not ask any more questions about it.

6. Example

In this section we provide a brief example of the way our analysis can be used. A complete version of this example is available [14].

We begin with a set of properties arranged in a taxonomy, as shown in Fig. 5. In Table 3 we provide some basic explanations of the intended meaning of these properties. The taxonomy we have chosen makes intuitive sense *prima facie*, and in most cases the taxonomic pairs were taken from existing ontologies such as Wordnet [24], Pangloss [20], and CYC [21]. See [10] for more similar examples of intuitive taxonomic orderings in existing ontologies that are inconsistent under our analysis.

The modeler, after making these initial decisions about the meta-properties and taxonomy, starts the system. For brevity, we assume the modeler enters the first four properties in batch mode, and the system responds, as shown in Fig. 6.

The final error indicates to the modeler that something is wrong with having *AMOUNT-OF-MATTER* subsume *LIVING-BEING*; the more general concept is mereologically extensional and has no unity, and a living being has biological unity. This is one of the most common modeling mistakes our methodology can reveal: living beings are not amounts of matter, they are *constituted* of matter. Constitution is not subsumption. The correction is to make *LIVING-BEING* subsumed directly by *ENTITY*.

The modeler makes this correction and proceeds. For the next property, *RED*, we show an interaction with the Q/A system in Fig. 7. In this interaction, we can see that the modeler is unsure about whether or not the property carries a UC, answering no to all questions. The system then

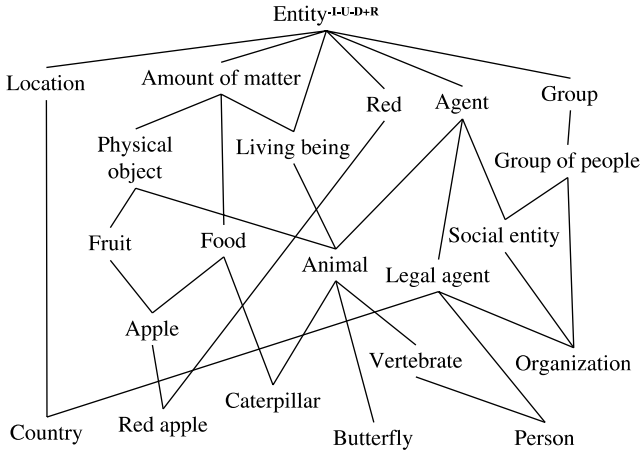

Fig. 5. A messy taxonomy.

Table 3
Meta-properties assigned to the initial properties

Property	Meta-properties	Kind	Notes
Entity	$-\mathbf{I}+\mathbf{U}-\mathbf{D}+\mathbf{R}$	Category	Everything is an entity
Location	$+\mathbf{O}\sim\mathbf{U}-\mathbf{D}+\mathbf{R}$	Type	Generalized regions of space. Locations are +ME (mereological extensional IC), no UC, there is no way to isolate a location
Amount-of-matter	$+\mathbf{O}\sim\mathbf{U}-\mathbf{D}+\mathbf{R}$	Type	Mass sortal: unstructured or scattered stuff, as lumps of clay or some bricks. +ME, no UC
Red	$-\mathbf{I}-\mathbf{U}-\mathbf{D}-\mathbf{R}$	Formal attribution	Intended as Red *thing*. No common IC nor UC/
Agent	$-\mathbf{I}-\mathbf{U}+\mathbf{D}\sim\mathbf{R}$	Formal role	Being an active participant in some event. Agents have no common IC/UC, i.e., one condition that holds for all instances
Group	$+\mathbf{O}\sim\mathbf{U}-\mathbf{D}+\mathbf{R}$	Type	Unstructured collections of wholes +ME. Note that $\sim U$ for groups represents a particular ontological choice by the modeler (which ignores the mathematical sense of groups)
Physical object	$+\mathbf{O}+\mathbf{U}-\mathbf{D}+\mathbf{R}$	Type	Isolated material objects. IC: same-spatial-location, topological UC
Living being	$+\mathbf{O}+\mathbf{U}-\mathbf{D}+\mathbf{R}$	Type	IC: same-DNA (necessary), biological UC
Group of people	$+\mathbf{I}-\mathbf{O}\sim\mathbf{U}-\mathbf{D}+\mathbf{R}$	Quasi-type	Groups whose elements are people. Like for group, IC is ME, when the people change it is a different group
Social entity	$-\mathbf{I}+\mathbf{U}-\mathbf{D}+\mathbf{R}$	Category	Groups of people brought together for some social reason. No *common* IC. UC: social connection
Fruit	$+\mathbf{O}+\mathbf{U}-\mathbf{D}+\mathbf{R}$	Type	Whole pieces of fruit. IC (necessary): same-plant and same-shape? topological UC
Food	$+\mathbf{I}-\mathbf{O}\sim\mathbf{U}+\mathbf{D}\sim\mathbf{R}$	Material role	Amounts of edible stuff. Dependent on things that eat them, nothing is food necessarily
Animal	$+\mathbf{O}+\mathbf{U}-\mathbf{D}+\mathbf{R}$	Type	IC: same-brain? biological UC
Legal agent	$+\mathbf{I}-\mathbf{U}+\mathbf{D}\sim\mathbf{R}$	Material role	Legally recognized entities. Local IC, no common unity, dependent on the legal body that recognizes them
Apple	$+\mathbf{O}+\mathbf{U}-\mathbf{D}+\mathbf{R}$	Type	IC (necessary): same shape, color, skin pattern? Topological UC
Country	$+\mathbf{I}+\mathbf{U}-\mathbf{D}\sim\mathbf{R}$	Phased sortal	Places, recognized by convention. IC: government, regions. Countries are countable, so some UC. A place can stop being country and still exist (e.g. Prussia)
Red apple	$+\mathbf{I}-\mathbf{O}-\mathbf{D}\sim\mathbf{R}$	Mixin	Inherits IC and UC from Apple. No apple is necessarily red
Caterpillar	$+\mathbf{I}+\mathbf{U}-\mathbf{D}\sim\mathbf{R}$	Phased sortal	IC: spots, same cocoon? Biological UC, but the same entity can be something else, so anti-rigid
Butterfly	$+\mathbf{I}+\mathbf{U}-\mathbf{D}\sim\mathbf{R}$	Phased sortal	IC: wing pattern? Biological UC, but the same entity can be something else, so anti-rigid
Vertebrate	$+\mathbf{I}-\mathbf{O}+\mathbf{U}-\mathbf{D}+\mathbf{R}$	Quasi-type	Vertebrate animals. Biological classification, adds membership conditions but probably no IC/UC
Organization	$+\mathbf{O}+\mathbf{U}-\mathbf{D}+\mathbf{R}$	Type	Groups of people together for some reason, with roles that define some structure. IC: same-mission (necessary), and functional UC
Person	$+\mathbf{O}+\mathbf{U}-\mathbf{D}+\mathbf{R}$	Type	IC could be same-fingerprint (sufficient), biological UC

```
---The property ENTITY does not carry unity
---Property ENTITY is a category.
---The property ENTITY does not carry identity and
is a non-sortal
---The property ENTITY is rigid
---The property ENTITY is independent
---The property ENTITY does not carry its own iden-
tity

Initial Classification of ENTITY: +CA +R -D -U -I -
O

---The property LOCATION does not carry unity
---Property LOCATION is a type.
---The property LOCATION is rigid
---The property LOCATION carries identity and is a
sortal
---The property LOCATION carries its own identity
---The property LOCATION is independent

Initial Classification of LOCATION: +I +R -D -U +O
+TP

---The property AMOUNT-OF-MATTER carries anti-
unity.
---The property AMOUNT-OF-MATTER does not carry
unity
---Property AMOUNT-OF-MATTER is a type.
---The property AMOUNT-OF-MATTER is rigid
---The property AMOUNT-OF-MATTER carries identity
and is a sortal
---The property AMOUNT-OF-MATTER carries its own
identity
---The property AMOUNT-OF-MATTER is independent

Initial Classification of AMOUNT-OF-MATTER: +I +O
~U -D +R -U +TP

---The property LIVING-BEING carries unity.
---Property LIVING-BEING is a type.
---The property LIVING-BEING is rigid
---The property LIVING-BEING carries identity and
is a sortal
---The property LIVING-BEING carries its own iden-
tity
---The property LIVING-BEING is independent
*CLASSIC ERROR* while processing:
  Trying to combine disjoint primitives:
    @tc{UC-PROP} and @tc{NON-UC-PROP}.
*EXPLANATION*: ~U (AMOUNT-OF-MATTER) cannot sub-
sume +U (LIVING-BEING).
```

Fig. 6. Sample output for first four properties in batch mode.

```
? (define-property 'red)
Is this property subsumed by any others? (y or n) y
What is it (list for multiple values): entity
Are instances of this property identified by a characteristic relation? (y or n) n
Is this property anti-rigid? (y or n) n
Does the property carry its own identity? (y or n) n
Is this property rigid? (y or n) n
Is this property non-rigid? (y or n) y
Are instances of this property dependent on instances of another property? (y or n) n
Are instances of this property independent? (y or n) y
Does the property carry an identity criterion (answer no if unknown)? (y or n) n
Are all instances of this property identifiable in different ways? (y or n) y
Does this property carry unity? (y or n) n
Are all instances of this property non-wholes? (y or n) n
Are instances of this property wholes under different relations? (y or n) n
Are all instances of this property countable? (y or n) n
Are there instances of this property that are not countable? (y or n) y
---The property RED does not carry unity
---Property RED is an attribution

Initial Classification of RED: -I -U -D -R -O +AT
```

Fig. 7. Sample output for *RED* in Q/A mode.

asks a question regarding countability, which is an indicator of unity. The modeler indicates that not all instances of red can be counted (consider the number of red patches in a red carpet), and the system concludes that the property does not carry unity.

This gives a flavor for how the systems works. We now skip to the next problematic property in the ontology. When the modeler enters the information for *PHYSICAL-OBJECT*, the system raises an error:

Trying to combine disjoint primitives: UC-PROP and NON-UC-PROP.

EXPLANATION: $\sim$ U (AMOUNT-OF-MATTER) cannot subsume

+ U (PHYSICAL-OBJECT).

This is yet another example of constitution being confused with subsumption. Physical objects are not themselves amounts of matter, they are constituted of matter. The solution is to make *PHYSICAL-OBJECT* subsumed directly by *ENTITY*.

The next problem occurs with the property *ANIMAL*, which was declared to be subsumed by *AGENT*:

```
Trying to combine disjoint primitives: INDEPENDENT-PROP
  and DEPENDENT-PROP. *
  EXPLANATION*: +D (AGENT) cannot subsume −D (ANIMAL).
```

This is a different kind of problem in which subsumption is being used to represent a type restriction. The modeler intends to mean, not that all animals are agents, but that animals *can be* agents. This is a very common misuse of subsumption, often used by object-oriented programmers. The correct way to represent this kind of relationship is with a covering, i.e., $\forall x\ AGENT(x) \rightarrow SOCIAL\text{-}ENTITY(x) \vee ANIMAL(x)$. Clearly this is a different notion than subsumption. The solution is to remove the subsumption link between *ANIMAL* and *AGENT*, and represent this information elsewhere.

The modeler proceeds, fixing problems like these until finished. The system uncovers three more errors regarding *BUTTERFLY*, *CATERPILLAR*, and *COUNTRY*. The former two are phased sortals, which require a common subsuming property according to the sortal expandability principle, so *LEPIDOPTERAN* is added to the taxonomy. The original choice of meta-properties for *COUNTRY* reveals that it is a phased sortal, and requires another phased sortal to "phase into". The modeler is forced to justify this decision and realizes that there are indeed two different properties: country as a social entity (*COUNTRY* in Fig. 8) and country as a location (*GEOGRAPHICAL REGION* in Fig. 8). The initial choice was based on a common misuse of multiple inheritance, representing multiple meanings of the same word.

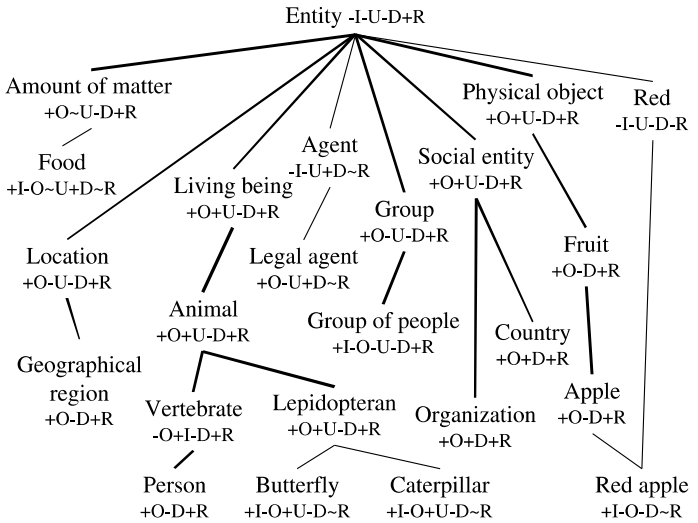

Fig. 8. The final taxonomy with highlighted backbone.

The final corrected taxonomy is shown in Fig. 8. More detailed descriptions of all the errors in the initial taxonomy and the solutions can be found in [14].

In addition to checking constraints and performing simple inference, the system also supports the methodology with several simple procedures for displaying useful slices of the ontology, such as the backbone taxonomy, the role taxonomy, phased sortals, etc.

Phased sortals are themselves cause for special consideration. Some attempt at describing them was made in [12] based on the work of Wiggins [32], however further analysis and clarification is needed. This remains an open issue. Real phased sortals seem to appear rarely in our experience, and therefore isolating them and checking that they are correct is a useful practice.

7. Conclusion

We have discussed several notions of formal ontology used for ontological analysis in Philosophy: identity, unity, essence, and dependence. We have formalized these notions in a way that makes them useful for conceptual modeling, and introduced a methodology for ontological analysis founded on these formalizations.

Our methodology is supported by a system that helps the conceptual modeler study the deep ontological issues surrounding the representation of properties in a conceptual model, and we have shown how this methodology can be used to analyze individual taxonomic links and make the taxonomy more understandable. In particular, we have also shown how to identify the backbone taxonomy, which represents the most important properties in an ontology that subsume every instance.

Unlike previous efforts to clarify taxonomies, our methodology differs in that:

- It focuses on the nature of the properties involved in subsumption relationships, not on the nature of the subsumption relation itself (which we take for granted).
- It is founded on formal notions drawn from ontology (a discipline centuries older than database design), and augmented with practical conceptual design experience, as opposed to being founded solely on the former or latter.
- It focuses on the validation of single subsumption relationships based on the *intended meaning* of their arguments in terms of the meta-properties defined here, as opposed to focusing on structural similarities between property descriptions.

Finally, it is important to note again that in the examples we have given, we are providing a way to make the *meaning* of properties in a certain conceptualization clear. We do not, for example, mean to claim that "Person is-a Legal-Agent" is wrong. We are trying to point out that *in a particular conceptualization* where *LEGAL-AGENT* has certain meta-properties (such as being anti-rigid) and *PERSON* certain others (such as being rigid), it is inconsistent to have person subsumed by legal-agent.

Acknowledgements

This work was supported in part by the Eureka Project (E! 2235) IKF, the Italian National Project TICCA (Tecnologie cognitive per l'interazione e la cooperazione con agenti artificiali), and a Research Committee Grant from Vassar College. We would like to thank Claudio Masolo,

Milena Stefanova, Pierdaniele Giaretta, Alessandro Oltramari and Bill Andersen for their useful comments.

References

[1] S. Bergamaschi, C. Sartori, On taxonomic reasoning in conceptual design, ACM Transactions on Database Systems 17 (3) (1992) 285–422.
[2] R. Brachman, What IS-A is and isn't: an analysis of taxonomic links in semantic networks, IEEE Computer 16 (10) (1983) 30–36.
[3] R.J. Brachman, D.L. McGuinness, P.F. Patel-Schneider, L. Resnick, A. Borgida, Living with CLASSIC: when and how to use a KL-ONE-like language, in: J. Sowa (Ed.), Principles of Semantic Networks, Morgan Kaufmann, Los Altos, CA, 1990, pp. 401–456.
[4] D. Calvanese, M. Lenzerini, D. Nardi, Description logics for conceptual data modeling, in: J. Chomicki, G. Saake (Eds.), Logics for Databases and Information Systems, Kluwer, Dordrecht, 1998, pp. 229–264.
[5] M. Cararra, P. Giaretta, Identity criteria and sortal concepts, in: C. Welty, B. Smith (Eds.), Formal Ontology in Information Systems, ACM Press, New York, 2001.
[6] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, Understanding top-level ontological distinctions, in: Proceedings of the 2001 IJCAI Workshop on Ontologies and Information Sharing, 2001.
[7] R.C. Goldstein, V.C. Storey, Data abstractions: why and how?, Data and Knowledge Engineering 29 (1999) 293–311.
[8] N. Guarino, M. Carrara, P. Giaretta, An ontology of meta-level categories, in: D.J.E. Sandewall, P. Torasso (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94), Morgan Kaufmann, San Mateo, CA, 1994, pp. 270–280.
[9] N. Guarino, Formal ontology in information systems, in: N. Guarino (Ed.), Formal Ontology in Information Systems. Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998, IOS Press, Amsterdam, 1998a, pp. 3–15.
[10] N. Guarino, The role of identity conditions in ontology design, in: Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends. Stockholm, Sweden, IJCAI, Inc, 1999b.
[11] N. Guarino, C. Welty, Identity, unity, and individuality: towards a formal toolkit for ontological analysis, in: Proceedings of ECAI-2000: The European Conference on Artificial Intelligence, IOS Press, Berlin, Germany, 2000.
[12] N. Guarino, C. Welty, A formal ontology of properties, in: R. Dieng (Ed.), Proceedings of 12th International Conference on Knowledge Engineering and Knowledge Management, Springer Verlag LNCS, Berlin, Germany, 2000.
[13] N. Guarino, C. Welty, Ontological analysis of taxonomic relations, in: A. Länder, V. Storey (Eds.), Proceedings of ER-2000: The International Conference on Conceptual Modeling, vol. 1920, Springer Verlag LNCS, Berlin, Germany, 2000.
[14] N. Guarino, C Welty, Ontology-driven conceptual analysis, in: AAAI-2000 Tutorial Presentation. Austin, Texas, 2000, Notes available at: http://www.cs.vassar.edu/faculty/welty/aaai-2000/.
[15] N. Guarino, C. Welty, Identity and subsumption, in: R. Green (Ed.), Semantic Relations, Kluwer, Dordrecht, 2001.
[16] G. Hirst, Existence assumptions in knowledge representation, Artificial Intelligence 49 (1991) 199–242.
[17] G.E. Hughes, M.J. Cresswell, A New Introduction to Modal Logic, Routledge, London, 1996.
[18] I.L. Humberstone, Intrinsic/extrinsic, Synthese 108 (1996) 205–267.
[19] A. Kaplan, Towards a consistent logical framework for ontological analysis, in: C. Welty, B. Smith (Eds.), Formal Ontology in Information Systems, ACM Press, New York, 2001.
[20] K. Knight, S. Luk, Building a large knowledge base for machine translation, in: Proceedings of American Association of Artificial Intelligence Conference (AAAI-94). Seattle, WA, 1994, pp. 773–778.
[21] D. Lenat, R.V. Guha, Building Large Knowledge-Based Systems, Addison-Wesley, Reading, MA, 1990.
[22] D. Lewis, New work for a theory of universals, Australasian Journal of Philosophy 61 (4) (1983).
[23] E.J. Lowe, Kinds of Being. A Study of Individuation, Identity and the Logic of Sortal Terms, Basil Blackwell, Oxford, 1989.
[24] G.A. Miller, WORDNET: a lexical database for english, Communications of ACM 2 (11) (1995) 39–41.
[25] J. McCarthy, Circumscription – A form of non-monotonic reasoning, Artificial Intelligence 13 (1980) 87–127.
[26] W.V.O. Quine, Ontological Relativity and Other Essays, Columbia University Press, New York, London, 1969.
[27] P. Simons, Parts: A Study in Ontology, Clarendon Press, Oxford, 1987.
[28] V.C. Storey, Understanding semantic relationships, Very Large Databases Journal 2 (1993) 455–488.
[29] V. Storey, D. Dey, H. Ullrich, S. Sundaresan, An ontology-based expert system for database design, Data and Knowledge Engineering 28 (1998) 31–46.
[30] P.F. Strawson, Individuals An Essay in Descriptive Metaphysics, Routledge, London and New York, 1959.
[31] R. Wieringa, W. De Jonge, P. Spruit, Roles and dynamic subclasses: a modal logic approach, in: Proceedings of European Conference on Object-Oriented Programming, Bologna, 1994.
[32] D. Wiggins, Sameness and Substance, Blackwell, Oxford, 1980.

Chris Welty is an Associate Professor at Vassar College, and has consulted in the real world at large and small companies including GE, AT&T, and IBM. He holds a Ph.D. in computer science from Rensselaer Polytechnic Institute. He is editor in chief of intelligence Magazine, steering committee chair of the Automated Software Engineering Conferences, an ACM Distinguished Lecturer, and the program chair of the 2001 conference on Formal Ontology in Information Systems (FOIS-2001). His research interests include ontology and ontological analysis, ontologies for information, for digital libraries, and for software understanding, and in general in improving information retrieval by representing knowledge.

Nicola Guarino is a senior research scientist at the Institute for System Theory and Biomedical Engineering of the Italian National Research Council (LADSEB-CNR). For about ten years now he has been actively promoting the study of the ontological foundations of knowledge representation and knowledge engineering with an interdisciplinary approach centered on logic, philosophy, and linguistics. He was chairman of the First International Conference on Formal Ontology in Information Systems (FOIS'98), is associate editor of the International Journal of Human and Computer Studies, has edited 3 journal special issues on ontology-related topics, and has published more than 30 papers in international journals, books and conferences. His research activities regard ontology design, conceptual modelling, knowledge sharing and integration, logical modeling of physical objects, and ontology-driven information retrieval.